John von Neumann Institute for Computing

Science Experimental Grid Laboratory (SEGL) Dynamical Parameter Study in Distributed Systems

N. Currle-Linde, U. Küster, M. Resch, B. Risio

published in

Parallel Computing:
Current & Future Issues of High-End Computing,
Proceedings of the International Conference ParCo 2005,
G.R. Joubert, W.E. Nagel, F.J. Peters, O. Plata, P. Tirado, E. Zapata (Editors),
John von Neumann Institute for Computing, Jülich,
NIC Series, Vol. **33**, ISBN 3-00-017352-8, pp. 49-56, 2006.

http://www.fz-juelich.de/nic-series/volume33

Science Experimental Grid Laboratory (SEGL) Dynamical Parameter Study in Distributed Systems

Natalia Currle-Linde[a], Uwe Küster[a], Michael Resch[a], Benedetto Risio[b]

[a]High Performance Computing Center Stuttgart (HLRS), University of Stuttgart, Nobelstrasse 19, 70569 Stuttgart, Germany

[b]RECOM Services, Nobelstrasse 15, 70569 Stuttgart, Germany

Currently, numerical simulation using automated parameter studies is already a key tool in discovering functional optima in complex systems such as biochemical drug design and car crash analysis. In the future, such studies of complex systems will be extremely important for the purpose of steering simulations. One such example is the optimum design and steering of high power furnaces of power plants. The performance of today's high performance computers and PC-clusters enables simulation studies with results that are as reliable as those obtained from physical experimentation. Recently, Grid technology has supported this development by providing uniform and secure access to computing resources over wide area networks (WANs), making it possible for industries to investigate large numbers of parameter sets using sophisticated simulations. However, the large scale of such studies requires organized support for the submission, monitoring, and termination of jobs, as well as mechanisms for the collection of results, and the dynamic generation of new parameter sets in order to intelligently approach an optimum. In this paper, we describe a solution to these problems which we call Science Experimental Grid Laboratory (SEGL). The system defines complex workflows which can be executed in the Grid environment, and supports the dynamic generation of parameter sets. It also allows the execution of sets of independent tasks of interdependent jobs which can run either synchronously or asynchronously on heterogeneous systems. The automatic collection of results is based on an object-oriented database design.

1. Introduction

During the last 20 years the numerical simulation of engineering problems has become a fundamental tool for research and development. In the past, numerical simulations were limited to a few specified parameter settings. Expensive computing time did not allow for more. More recently, computer clusters with hundreds of processors enable the simulation of complete ranges of multi-dimensional parameter spaces in order to predict an operational optimum for a given system. Testing the same program in hundreds of individual cases may appear to be a straightforward task. However, the administration of a large number of jobs, parameters and results poses a significant problem. An effective mechanism for the solution of such parameter problems can be created using the resources of a Grid environment. This paper, furthermore proposes the coupling of these Grid resources to a tool which can carry out the following: generate parameter sets, issue jobs in the Grid environment, control the successful operation and termination of these jobs, collect results, and generate new parameter sets based on previous results in order to approach a functional optimum, after which the mechanism should gracefully terminate. We expect to see the use of parameterized simulations in many disciplines. Examples are drug design, statistical crash simulation of cars, airfoil design, power plant simulation by varying burners and fuel quality. The mechanism proposed here offers a unified framework for such large-scale optimization problems in design and engineering.

1.1. Existing tools for parameter investigation studies

Tools like Nimrod [1] and Ilab [1] enable parameter sweeps and jobs, running them in a distributed computer environment (Grid) and collecting the data. ILab also allows the calculation of multi-parametric models in independent separate tasks in a complicated workflow for multiple stages. However, none of these tools is able to dynamically generate new parameter sets by an automated optimization strategy. In addition to the above mentioned environments, tools like Condor [1], UNICORE [3] or AppLeS [1] can be used to launch pre-existing parameter studies using distributed resources. These, however, give no special support for dynamic parameter studies.

1.2. Workflow

Realistic application scenarios become increasingly complex due to the necessary support for multiphysics applications, preprocessing steps, postprocessing filters, visualization, and the iterative search in the parameter space for optimum solutions. These scenarios require the use of various computer systems in the Grid, resulting in complex procedures best described by a workflow specification. The definition and execution of these procedures requires user-friendly workflow description tools with graphical interfaces, which support the specification of loops, test and decision criteria, synchronization points and communication via messages. Several Grid workflow systems exist. Systems such as Triana [4] and UNICORE, which are based on directed acyclic graphs (DAG), are limited with respect to the power of the model; it is difficult to express loop patterns, and the expression of process state information is not supported. On the other hand, workflow-based systems such as GSFL [6], and BPEL4WS [6] have solved these problems but are too complicated to be mastered by the average user. With these tools, even for experienced users, it is difficult to describe non-trivial workflow processes involving data and computing resources. The SEGL system described here aims to overcome these deficiencies and to combine the strengths of Grid environments with those of workflow oriented tools. It thus provides a visual editor and a runtime workflow engine for dynamic parameter studies.

1.3. Dynamic parameterization

Complex parameter studies can be facilitated by allowing the system to dynamically select parameter sets on the basis of previous intermediate results. This dynamic parameterization capability requires an iterative, self-steering approach. Possible strategies for the dynamic selection of parameter sets include genetic algorithms, gradient-based searches in the parameter space, and linear and nonlinear optimization techniques. An effective tool requires support of the creation of applications of any degree of complexity, including unlimited levels of parameterization, iterative processing, data archiving, logical branching, and the synchronization of parallel branches and processes. The parameterization of data is an extremely difficult and time-consuming process. Moreover, users are very sensitive to the level of automation during application preparation. They must be able to define a fine-grained logical execution process, to identify the position in the input data of parameters to be changed during the course of the experiment, as well as to formulate parameterization rules. Other details of the parameter study generation are best hidden from the user.

1.4. Databases

The storage and administration of parameter sets and data for an extensive parameter study is a challenging problem, best handled using a flexible database. An adequate database capability must support the *a posteriori* search for specific behavior not anticipated in the project. In SEGL the automatic creation of the project and the administration of data are based on an object-oriented database (OODB) controlled by the user. The database collects all relevant information for the

realization of the experiment, such as input data for the parameter study, parameterization rules and intermediate results. In this paper we present a concept for the design and implementation of SEGL, an automated parametric modeling system for producing complex dynamically-controlled parameter studies.

2. System Architecture and Implementation

Figure 1 shows the system architecture of SEGL. It consists of three main components: the User Workstation (Client), the ExpApplicationServer (Server) and the ExpDBServer (OODB). The system operates according to a Client-Server-Model in which the ExpApplication Server interacts with remote target computers using a Grid Middleware Service. The implementation is based on the Java 2 Platform Enterprise Edition (J2EE) specification and JBOSS Application Server. The System runs on Windows as well as on UNIX platforms. The OODB is realized using the Java Data Objects (JDO) implementation of FastObjects [5].

The client on the user's workstation is composed of the ExpDesigner and the ExpMonitorVIS. The ExpDesigner is used to design, verify and generate the experiment's program, organize the data repository and prepare the initial data. The ExpMonitorVIS is generated for visualization and for the actual control of the complete process. The ExpDesigner allows to describe complex experiments using a simple graphical language. Each experiment is described at three levels: control flow, data flow and data repository. The control flow level is used for the description of the logical schema of the experiment. On this level the user makes a logical connection between blocks: direction, condition and sequence of the execution of blocks. Each block can be represented as a simple parameter study.

The data flow level is used for the local description of interblock computation processes. The description of processes for each block is displayed in a new window. The user is able to describe:

(a) Both a standard computation module and a user-specific computation module. The user-specific module can be added to suit the application domain.

(b) The direction of input and output data between the metadata repository and the computation module.

(c) The parameterization rules for the input set of the data.

(d) Synchronization of interblock processes.

On the data repository level, a common description of the metadata repository is created. The repository is an aggregation of data from the blocks at the data flow level. Each block contains one or more windows representing part of the data flow. Also described at the data repository level are the key and service fields (objects) of the data base.

After completion of the design of the program at the graphical icon-level, it is "compiled". During the "compilation" the following is created:

(a) a table of the connections between program objects on the data flow level for each block (manipulation of data) and

(b) a table of the connections between program blocks on the control flow level for the experiment.

Parallel to this, the experiment's database aggregates the data base icon objects from all blocks / windows at the data flow level and generates query-language (QL) descriptions of the experiment's database. The container application of the experiment is transferred to the ExpApplicationServer and the QL descriptions are transferred to the server data base. Here, the metadata repository is created. The ExpApplicationServer consists of the ExpEngine, the Task, the ExpMonitorSupervisor and the ResourceMonitor.

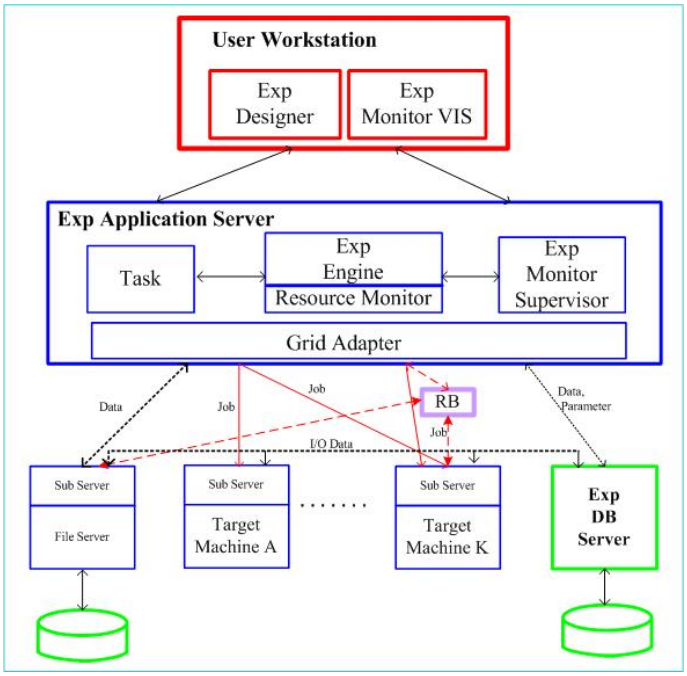

Figure 1: System Architecture

The Task is the container application. The ResourceMonitor holds information about the available resources in the Grid environment. The MonitorSupervisor controls the work of the runtime system and informs the Client about the current status of the jobs and the individual processes. The ExpEngine is the controlling subsystem of SEGL (Runtime subsystem). It consists of three subsystems: the TaskManager, the JobManager and the DataManager. The TaskManager is the central dispatcher of the ExpEngine coordinating the work of the DataManager and the JobManager:

(1) It organizes and controls the sequence of execution of the program blocks. It starts the execution of the program blocks according to the task flow and the condition of the experiment program.

(2) It activates a particular block according to the task flow, chooses the necessary computer resources for the execution of the program and deactivates the block when this section of the program has been executed.

(3) It informs the MonitorSupervisor about the current status of the program.

The DataManager organizes data exchange between the ExpApplicationServer and the FileServer and between the FileServer and the ExpDBServer. Furthermore, it controls all parameterization processes of input data. The JobManager generates jobs and places them in the corresponding SubServer of the target machines. It controls the placing of jobs in the queue and observes their execution.

The final component of SEGL is the data base server (ExpDBServer). All data which occurred during the experiment, initial and generated, are kept in the ExpDBServer. The ExpDBServer also hosts a library tailored to the application domain of the experiment. For the realization of the data base we choose an object-oriented database because its functional capabilities meet the requirements of an information repository for scientific experiments. The interaction between ExpApplicationServer and the Grid resources is done through a Grid Adaptor. Currently, e.g. Globus[2] and UNICORE offer these services.

3. Parameter Modeling from the user's view

Figure 2 shows an example of a task flow for an experiment as it appears in the ExpDesigner. The graphical description of the application flow has two purposes: first, it is used to collect all information for the creation of the experiment and, second, it is used for the visualization of the current experiment in the ExpMonitorVIS. For instance, the current point of execution of a computer process is highlighted in a specific color within a running experiment.

3.1. Control Flow level

Within the control flow (see Figure 2) the user defines the sequence of the execution of the experimentâs blocks. There are two types of operation block: control block and solver block. The solver block is the program object which performs some complete operation. The standard exam-

ple of the solver block can be a simple parameter sweep. The control block is the program object which allows the changing of the sequence of execution operation according to a specified criterion.

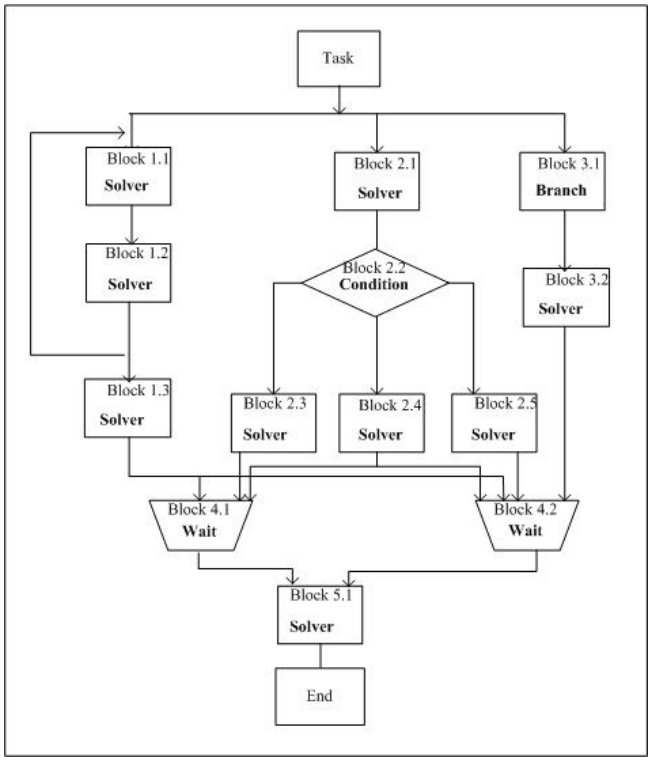

Figure 2: Sample Task Flow (control flow)

Figure 2 shows an example of task flow. After execution of "Task" block 1.1, block 2.1 and block 3.1 are activated simultaneously. In each of these blocks a process is executed. After having worked with the first set of data in block 1.1, the first process in block 1.2 is activated. After execution of the first process in block 1.2, the first process in block 1.3 and the second process in block 1.1 are started according to the logic of the experiment. The input data for the second and the following processes in block 1.1 are prepared in block 1.2 and so on.

3.2. Data Flow level

Figure 3 presents an example of a solver block (Block 1.1). At this level, the user can describe the manipulation of data in a very fine grained way. The solver block consists of computation (C) , replacement (R), parameterization (P) modules and a data base. These are connected to each other with arrowed lines showing the direction of data transfer between modules and the sequence of execution during the computation process.

Each module is a Java object, which has a standard structure and consists of several sections. For example: each computation module (C) consists of four sections. The first section organizes the preparation of input data. The second generates the job and controls its execution. The third initializes and controls the record of the result in the experiment data base. The fourth section controls the execution of module operation. It also informs the main program of the block about the manipulation of certain sets of data and when execution within a block is complete.

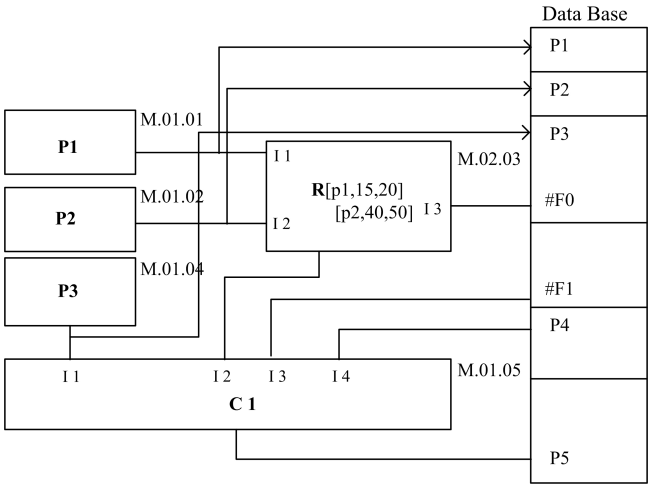

Figure 3: Solver Block 1.1 (data flow)

After a block is started, the parameterization module (P) and replacement module (R) wait for the request from the corresponding inputs of the computation module (C). After that, they generate a set of input data according to rules specified by the user, either as mathematical formulae or a list of parameter values. In this example three variants of parameterization are represented:

(a) Direct transmission of the parameter values with the job. In this case, parameterization module (P3) transfers the generated parameter value to the computation module (C1) upon its request. The computation module generates the job, including converting parameter values into corresponding job parameters. This

method can be used if the parameterized value is a number, symbol or combination of both.

(b) Parameterized objects are large arrays of information (DB-P4 in Figure 3) which are kept in the experiment data base. These parameters are copied directly from the experiment data base to the corresponding file server and then written with the same array name with the index of the number of the stage. In this case, attributes of the job are sent to the file server as references (an array of data).

(c) If it is important, then the preparation of the data is moved outside of the main program. This allows the creation of a more universal computation module. Furthermore, it allows scaling, i.e. avoiding limitations in the size, position, type and number of the parameterized objects used in a module.

In these cases the replacement module is used. During the preparation of the next set of input data, new parameter values P1 und P2 are generated. The generated parameter set is linked with replacement processes and then delivered to the corresponding FileServer, where the replacement process is executed.

After the replacement of the specified parameters, the input data is ready for the first stage of computation. Computation module C1 sends a message to the JobManager to prepare the job for the first stage. The JobManager chooses the computer resources currently available in the network and starts the job. After confirmation from the corresponding SubServer of the Target Machine that the job is in a queue, the preparation of the next set of data for the next computation stage begins. Each new stage carries out the same processes as the previous stage. At all stages, the output file is archived immediately after being received by the experiment's database. The control of all processes takes place according to the pattern described above. After starting the ExpMonitorVIS on their workstation, the user receives continuously updated status information regarding the experiment's progress.

4. Use case: Power plant simulation by varying burners and fuel quality

The liberalization of the energy markets puts more and more pressure on the competitiveness of power companies throughout the world. In order to maintain their competitive edge, it is necessary to optimize the operation of existing power plants towards minimum operational costs. Potential optimization targets can be minimization of excess air (increasing efficiency) or NOx-emission (reducing DeNOx operation costs). Pure experimental optimizations without computer-aided techniques are time-consuming and require a significantly higher manpower effort. Furthermore, in the case of necessary design changes the technical risks involved in the investment decision can only be assessed with computer-aided techniques. Computer-aided methods are well accepted in the power industry. The optimization procedure applied by SEGL for the present problem is based on a genetic algorithm (GA).

In order to work on boiler optimization problems with SEGL, the parameters that have to be optimized are coded in binary form and assembled to a so-called "chromosome". The chromosome carries all the important properties to be changed of the so-called "individuals". A certain number of these artificial individuals are generated initially, the so-called "population", and the GA of SEGL imitates the natural evolution process. The imitation is done by applying the genetic mechanisms Selection, Recombination and Mutation. The basic workflow can be described as follows:

1. Binary coding of optimization parameters and chromosome assembly.
2. Generation of an initial population.
3. Decoding of the chromosome information for each individual.
4. Simulation of the decoded set of optimization parameters with the 3D-furnace simulation code RECOM-AIOLOS for each individual. This is the time consuming step.

5. Filtering the 3-D results of the furnace simulation to derive the target values for each individual.

6. Evaluation of the performance level for each individual (terminate the optimization process if desired optimization level is reached).

7. Selection of suitable individuals for reproduction and Recombination/Mutation of the chromosome information for the selected individuals to generate new individuals.

8. Return to Step 3 for new individuals.

4.1. Industrial Applicability

An experimental operation optimization exercise performed in 1991 at a power station in Italy (ENEL's coal-fired Fusina) is used to demonstrate the capabilities of SEGL. In a windbox, the amount of air flowing through a nozzle is controlled by the damper setting of the nozzle. A damper setting of 100% means that the flow passage of the nozzle is fully open. Reducing the damper setting of a single nozzle allows the reduction of the air mass flow through the nozzle, but at the same time the air mass flows for all other nozzles in the windbox are increased. In 1991 separate overfire air nozzles (separate OFA) were installed above the main combustion zone to minimize NOx-emissions. A new operation mode was required after the successful installation of the separate overfire air to maintain the lowest possible NOx-emission together with a minimum unburned carbon loss. In 1991 this optimization exercise was solved experimentally. In a series of 15 tests over a duration of approximately 10 days, 15 operation modes were tested with varying amounts of close coupled overfire air (CCOFA), separate OFA, and tilting angle of the separate OFA ($\pm 30^o$).

The following operation experience was recorded to identify an optimized operation:

a) For a horizontal orientation of the separate OFA the maximum NOx-reduction is reached with dampers 100% open.

b) A tilting of the separate OFA to -30^o has a minor effect on the NOx-emission but improves the burnout (reduced unburned carbon loss).

c) A tilting of the separate OFA to $+30^o$ leads to an NOx-reduction but increases the unburned carbon loss significantly.

d) Closing the CCOFA completely at 100% open separate OFA has only a minor effect on the NOx-emission.

In order to work on this combustion optimization problem in virtual reality, a high-resolution boiler model with 1 Mio. grid points and a reduced boiler model with only 200,000 grid points was generated. In order to reduce the computational effort, the optimization environment works only with the reduced boiler model. The spatial resolution of the reduced boiler model was reduced to a degree that still allows qualitatively accurate predictions. For the optimized settings that are identified during the automatic optimization, a simulation run on the high-resolution model is required to generate quantitatively reliable answers (see Table 1). As shown in Table 1, an accuracy of approximately $\pm 10\%$ between simulation and reality can be reached on the high-resolution boiler model. The optimization parameters "OFA damper setting", "CCOFA damper setting", and "Tilting Angle" were coded with 4 bit on the chromosomes. NOx-emission and C in Ash values achieved in the model were combined to a target function for the evaluation of the individuals. The underlying combined evaluation target function is

Target Function = Evaluation [NOx] + Evaluation [C in Ash].

The GA required approximately 11 generations with 10 individuals per population to identify an optimized parameter set. During the course of the automatic optimization, approximately 51 of the entire 4096 ($2^4 \cdot 2^4 \cdot 2^4$) coded combinations of parameter settings were evaluated with respect to the target functions. Table 2 shows the development of the best individuals in each generation in the course of the automatic optimization. The results demonstrate that SEGL is able to identify

Table 1
Measured and calculated (high-resolution) NOx-emission and C in Ash

	NOx-emission $[mg/m_n^3, 6\%O2]$		C in Ash [%]	
Setting	measured	calculated	measured	calculated
No OFA No CCOFA	950 - 966	954	6.41 - 7.50	5.66
No OFA CCOFA: 100%	847 - 858	794	7.47 - 7.61	6.58
OFA:100% CCOFA: 100%	410 - 413	457	10.43 - 11.48	10.28

the same positive measures that were found in the experimental optimization. The final run on the high-resolution boiler model led to an NOx-emission of $476mg/m_n^3$ at $6\%O_2$ and a C in Ash value of 8.42 %. Both values are in the range of the emission and C in Ash values that were observed in the field after the optimization exercise. The total duration of the automatic optimization was only 3.5 days on a high performance vector-computer.

Table 2
Development of best individuals in each generation during automatic optimization

Generation	Target-Value	OFA [%]	CCOFA [%]	Tilting Angle $[^o]$	NOx mg/m_n^3	C in Ash[%]
Basis	12.070	0	0	0	805	3.39
1	10.061	100	100	-30	479	10.84
5	9.600	93	93	-30	473	10.42
10	9.177	93	20	-30	458	10.26

5. Conclusion

This paper presented the concept and description of the implementation of SEGL for the design of complex and hierarchical parameter studies which offers an efficient way to execute scientific experiments. We can show that SEGL allows to substantially reduce optimization costs for parameter studies.

References

[1] de Vivo, A., Yarrow, M. McCann, K.: A comparison of parameter study creation and job submission tools; Technical report NAS-01002, NASA Ames Research Center, Moffet Filed, CA, 2000

[2] Foster, I., Kesselman C.: The Globus Project: A Status Report; In: Proc. IPPS/SPDP'98 Heterogeneous Computing Workshop, pp4-18, 1998.

[3] Erwin, D. (Ed.): Joint Project Report for the BMBF Project UNICORE Plus, Grant Number: 01 IR 001 A-D, Duration: January 2000 - December 2002;

[4] Taylor, I., Shields, M., Wang, I., and Philp, R.: Distributed P2P Computing within Triana: A Galaxy Visualization Test Case; IPDPS 2003 Conference, Nice / France, 2003.

[5] FastObject webpage: http://www.fastobjects.com

[6] Tony, A., Curbera, F., Dholakia, H., Goland, Y., Klein, J., Leymann, F., Liu, K., Roller, D., Smith, D., Thatte, S., Trickovic, I., Weerawarana, S.: Specification: Business Process Execution Language for Web Services Version 1.1, May 05, 2003: http://www.106.ibm.com/developerworks/library/ws-bpel